PSO算法加速因子的非线性策略研究

陈水利，蔡国榕（集美大学理学院，福建 厦门 361021；福州大学数学与计算机科学学院，福建 福州 350002）
郭文忠，陈国龙（福州大学数学与计算机科学学院，福建 福州 350002）

[摘要] 为了有效地控制PSO算法的全局和局部搜索能力，着重分析了PSO算法中加速因子对粒子收敛的影响，提出加速因子采用反余弦非对称策略能有效提高PSO算法的搜索性能。对5种加速因子策略使用4个著名的基准函数进行测试，试验结果表明，反余弦非对称方法可以使粒子在搜索的初期获得更好的多样性，在算法后期则可以有效增强粒子的搜索能力，从而使算法具有更强的摆脱局部极值的能力，提高算法的收敛精度。

[关键词] 粒子群优化算法（PSO）；加速因子；非线性
[中图分类号] TP301.6 [文献标识码] A [文章编号] 1673-1409 (2007) 04-N001-04

粒子群优化（particle swarm optimization，PSO）算法❶是由美国的Kennedy和Eberhart受鸟群觅食行为的启发于1995年提出的。该算法是一种基于群体的、具有全局寻优能力的优化工具，通过群体中粒子间的合作与竞争产生的群体智能指导优化搜索，每代种群中的解具有“自我”学习提高和向“他人”学习的双重优点。目前，PSO算法已逐渐成为一个研究热点，被广泛应用于多目标优化、神经网络训练、模糊系统控制和决策支持等领域。

粒子群算法的加速因子反映了粒子对自身和社会信息的一种平衡，对粒子的飞行范围和飞行速度起到了关键作用。为了找到一种能在全局搜索和局部搜索之间取得最佳平衡的参数选取方法，研究人员进行了大量的工作，先后提出了线性变化策略❷、模糊控制策略❸❹和随机策略❺等，这些方法旨在丰富PSO算法种群的多样性，扩大算法的搜索范围，增强算法的爬坡能力。

笔者主要从加速因子的取值对算法性能的影响入手，分析加速因子对粒子运动方式和运动轨迹的影响，从而提出科学可行的自适应调整策略。在PSO算法初期应加强社会知识的搜索，初期要使粒子尽可能的飞跃整个搜索空间，以期获得粒子的多样性；而在搜索末期，粒子则应该保持一定的速度，从而尽可能摆脱局部极值的干扰。基于上述思想，笔者提出采用非线性调整（反余弦）非对称策略。

1 标准PSO算法

PSO算法强调群体性、协作性，该算法初始化为一群D维空间的随机粒子（随机解）。然后通过迭代找到最优解。在每一次迭代中，粒子通过跟踪2个“极值”来更新自己。第一个就是粒子本身所找到的最优解p_{best}，另一个极值是整个种群目前找到的最优解g_{best}。找到这2个最优值后，粒子根据如下的公式来更新自己的速度和新的位置：

[收稿日期] 2007-09-23
[基金项目] 国家自然科学基金项目（10471083）；教育部科学技术研究重点基金项目（206073）；福建省自然科学基金项目（A0610012）；福建省科技厅重点项目（2004H007）。
[作者简介] 陈水利（1956-），男，1978年大学毕业，教授，现主要从计算智能、模糊信息处理等方面的教学与研究工作。

❶ Kennedy J, Eberhart R. Particle swarm optimization. International Conference on Neural Networks, 1995. 1942～1948.
❷ Shi Y, Eberhart R. Empirical study of particle swarm optimization. International Conference on Evolutionary Computation, 1999. 1945～1950.
❸ Shi Y, Eberhart R. Fuzzy adaptive particle swarm optimization. The IEEE Congress on Evolutionary Computation, 2001. 101～106.
❹ Shi Y H, Eberhart R C. Particle Swarm optimization with fuzzy adaptive inertia weight. IEEE Proceedings of the Workshop on Particle Swarm Optimization, 2001.
❺ Eberhart R, Shi Y. Tracking and optimizing dynamic systems with particle swarms. The IEEE Congress on Evolutionary Computation, 2001. 94～100.

$$v_{id}(t+1) = wv_{id}(t) + c_1\text{rand}()(p_{id} - x_{id}(t)) + c_2\text{rand}()(p_{gd} - x_{id}(t)) \quad (1)$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \quad (2)$$

式中，$x_i(t) = (x_{i1}(t), x_{i2}(t), \cdots, x_{iD}(t))$ 表示微粒 i 在 t 时刻的位置；$v_i(t) = (v_{i1}(t), v_{i2}(t), \cdots, v_{iD}(t))$ 表示粒子 t 时刻的速度；$p_i = (p_{i1}, p_{i2}, \cdots, p_{iD})$ 表示粒子 i 到达过的最好的位置；$p_g = (p_{g1}, p_{g2}, \cdots, p_{gD})$ 表示当前的全局最优点；w 为惯性权重(inertia weight)；c_1 和 c_2 为加速因子(acceleration coefficient)；rand()和 rand()为 2 个在[0,1]范围内变化的随机函数。PSO 使用公式(1)、(2)反复改变粒子的速度和位置，直到达到最大循环次数或终止条件为止，此时的全局最优解为最终结果。一般情况下，迭代中止条件选为最大迭代次数或粒子群迄今为止搜索到的最优位置满足适应阈值。

2 加速因子的非线性策略

在PSO算法中，因为加速因子 c_1 和 c_2 决定了粒子本身经验信息和其他粒子的经验信息对粒子运动轨迹的影响，反映了粒子群之间的信息交流。设置较大的 c_1 值，会使粒子过多的在局部徘徊；相反地，较大的 c_2 值会使粒子过早收敛到局部最优值。在理想状态下，搜索初期要使粒子尽可能地飞跃整个搜索空间，以期获得粒子的多样性；而在搜索末期，粒子则应该保持一定的速度，从而尽可能摆脱局部极值的干扰。

传统的 PSO 算法通过设置固定的加速因子使得算法在全局和局部的搜索能力之间达到最佳平衡，针对不同的问题背景，c_1 和 c_2 一般取值为 1.0～2.5。文献[1]提出利用线性调整加速因子取值，即 c_1 先大后小，而 c_2 先小后大的思想，其原理是在算法运行初期个体飞行速度主要参考本身信息，而后期则更加注重社会(群体)知识。该方法在某些应用背景上得到较好的效果，但也存在粒子易过早收敛于局部极值的缺点，这主要是因为在搜索的前期过大的 c_1 和过小的 c_2 使得粒子在全局徘徊，而后期阶段粒子缺乏多样性，导致过早的收敛到局部最优。为了解决上述问题，文献[2]提出可以用凹函数的策略调整加速因子，该方法在算法早期加快 c_1 和 c_2 的变化速度，目的是让算法较快地进入局部搜索，试验结果表明该方法是可行的，但是改进效果不是很明显，原因在于算法后期过小的 c_1 和过大的 c_2 使得群体中粒子在位置上缺乏多样性，即没有从根本上解决线性调整策略存在的问题。

笔者采用反余弦函数来构造加速因子调整策略，如图 1。反余弦策略的特点在于初期通过加速 c_1 和 c_2 的变化使得算法较快地进入局部搜索，在算法后期则设置比线性和凹函数策略更理想的 c_1 和 c_2 值，使粒子保持一定的搜索速度，避免过早收敛。反余弦加速因子构造方式具体如下：

$$c_1 = c_{1\min} + (c_{1\max} - c_{1\min})\left(1 - \frac{\arccos(\frac{-2 \times \text{CurrentIterances}}{\text{MaxIterances}} + 1)}{\pi}\right) \quad (3)$$

$$c_2 = c_{2\max} - (c_{2\max} - c_{2\min})\left(1 - \frac{\arccos(\frac{-2 \times \text{CurrentIterances}}{\text{MaxIterances}} + 1)}{\pi}\right) \quad (4)$$

其中，CurrentIterances为当前迭代次数；MaxIterances表示最大迭代数；$c_{1\max}$、$c_{2\max}$ 分别为 c_1 和 c_2 的迭代终值；$c_{1\min}$、$c_{2\min}$ 分别为 c_1、c_2 的初始值。

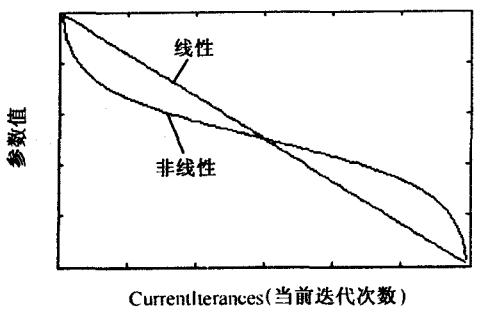

图 1 反余弦加速因子 c_1

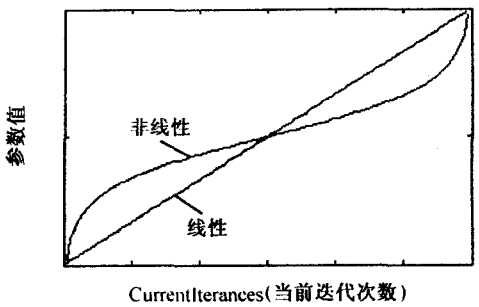

图 2 反余弦加速因子 c_2

对于 c_1、c_2 的参数设置范围，Ratnaweera 等中使用线性对称的的方法进行了试验[1]。模拟结果表明，当 c_1 由 2.5 线性递减至 0.5，c_2 由 0.5 线性递增至 2.5 时，算法所获得适应值较优。该方法确实加速了算法的收敛，尤其在单峰值函数的测试中表现突出。为此付出的代价是算法容易陷入局部最小值，在多峰值函数的测试中容易过早收敛。针对此问题，文献[3]通过多种参数组合进行验证表明：c_1、c_2 采用非对称的变化范围($c_1 = 2.75 \sim 1.25$，$c_2 = 0.50 \sim 2.25$)，可以得到比上述对称方法更优的结果。

因此，可以采用非对称、反余弦方法对加速因子进行动态调整。测试结果也表明了新的加速因子调整设置 PSO 算法对单峰及部分多峰函数的收敛性具有明显的改进，并且较原先的设置具有更强的全局搜索能力，这也说明了笔者提出的策略拥有更强的追踪粒子群中全局最优粒子的能力和发现全局最优解的能力。

3 试　　验

3.1 试验设置

使用 4 个著名的基准函数来测试新算法设置的性能。其中，De Jong 函数是单峰函数，其全局最小值为 0，收敛于 (0，0，…，0)；Rosenbrock、Rastrigrin、Griewank 函数❶❷均是经典的多峰函数，全局最小值为 0，收敛于 (0，0，…，0)。测试函数的具体数学描述如表 1 所示。

表 1　4 个测试函数

函　数	函数表达式	维数 D	搜索空间	最大速度
De Jong 函数	$f_1 = \sum_{i=1}^{D} x_i^2$	30	$(-100, 100)^D$	100
Rosenbrock 函数	$f_1 = \sum_{i=1}^{D} [100 \times (x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	$(-100, 100)^D$	100
Rastrigrin 函数	$f_1 = \sum_{i=1}^{D} [x_i^2 - 10 \times \cos(2\pi x_i) + 10]$	30	$(-10, 10)^D$	10
Griewank 函数	$f_1 = \frac{1}{4000} \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos \frac{x_i}{\sqrt{i}} + 1$	30	$(-600, 600)^D$	600

考虑用 5 种加速因子策略进行测试：方法 1 为标准 PSO ($c_1 = c_2 = 2$)；方法 2 为线性变换策略 PSO($c_1 = 2.5 \sim 0.5$，$c_2 = 0.5 \sim 2.50$)；方法 3 为凹函数策略[7]($c_1 = 2.5 \sim 0.5$，$c_2 = 0.5 \sim 2.5$)；方法 4 为反余弦策略($c_1 = 2.5 \sim 0.5$，$c_2 = 0.5 \sim 2.50$)；方法 5 为非对称反余弦策略($c_1 = 2.75 \sim 1.25$，$c_2 = 0.50 \sim 2.25$)。在算法执行过程中，惯性权值 w 的取值范围随着算法迭代次数的增加从 0.4 线性变化到 0.9[8]，每个基准函数的维数为 30，粒子数为 40，最大迭代次数为 1500 代，每个设置重复运行 50 次。图 3～6 分别给出了 4 个基准函数 50 次试验平均最优解随迭代次数的变化历程。性能评估采用如下方法：①跟踪算法运行过程中适应值的变化，评估作为算法收敛速度和收敛精度；②与参考文献报道的性能进行比较。

3.2 结果与分析

5 种不同方法对 4 个标准测试函数的运算结果如表 2。

1) DeJong 函数　从表 2 可以看出，反余弦加速因子方法的收敛精度较线性变化和凹函数策略有明显提高。图 3 则表明了该方法可以有效避免由于粒子发散而造成的收敛失败，而采用 $c_1 = 2.75 \sim 1.25$，$c_2 = 0.50 \sim 2.25$ 的收敛精度略优于对称变化方法。

2) Rosenbrock 函数　表 2 说明了加速因子采用非线性递归策略较其他方法有一定的提高。图 4 可

❶ Shi Yu hui, Eberhart R. Parameter selection in particle swarm optimization. IEEE Proc of the 7th Annual Conf on Evolutionary Programming, 1998. 591～600.

❷ Angeline P J. Using Selection to Improve Particle Swarm Optimization. IEEE International Conference on Evolutionary Computation, 1998.

以看出凹函数策略、反余弦策略初期收敛速度比线性策略快，这与文献［1］所得到的结论是一致的。而笔者所提出的方法相比凹函数在多峰问题上则有进一步的改进，尤其在算法后期，反余弦方法由于粒子保留比凹函数策略更好的搜索能力，收敛精度得到有效提高。

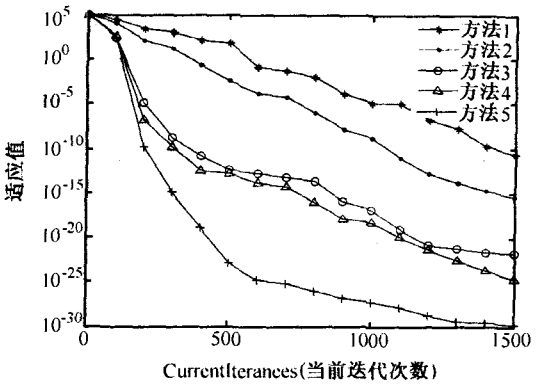

图 3 De Jong 函数优化的迭代曲线

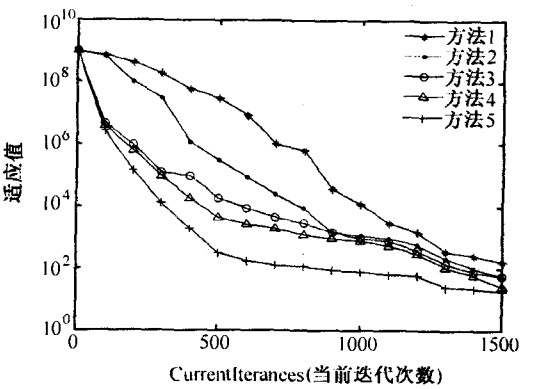

图 4 Rosenbrock 函数优化的迭代曲线

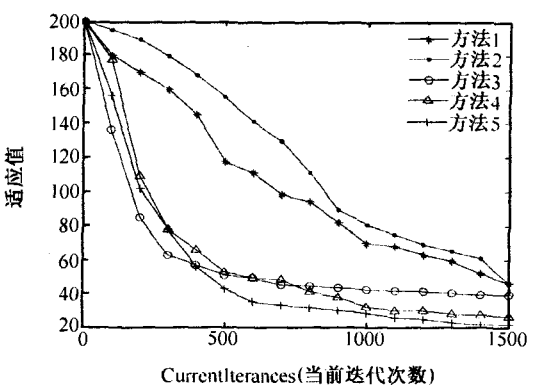

图 5 Rastrigrin 函数优化的迭代曲线

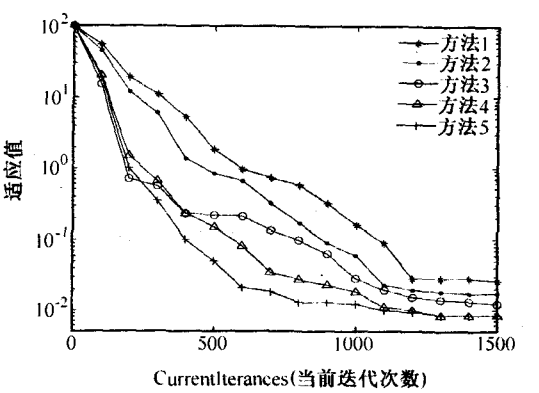

图 6 Griewank 函数优化的迭代曲线

表 2 利用 5 种不同方法对 4 个标准测试函数运算结果

统计项	函数											
	De Jong 函数			Rosenbrock 函数			Rastrigrin 函数			Griewank 函数		
	最优值	均 值	标准差	最优值	均 值	标准差	最优值	均 值	标准差	最优值	均 值	标准差
方法 1	1.01447e-016	2.20604e-011	1.14362e-010	1.11648	160.798	158.159	22.8845	46.099	17.742	7.77156e-016	0.025758	0.022675
方法 2	1.00033e-025	4.12063e-016	1.56517e-015	4.22094	55.309	52.672	21.8891	59.211	21.873	1.66533e-015	0.0172441	0.0166038
方法 3	4.97683e-024	2.58839e-022	1.42522e-019	0.01474	54.512	48.772	18.9042	39.206	11.581	0	0.0120479	0.0138814
方法 4	1.41841e-028	1.92477e-025	1.07094e-024	0.00620	25.271	28.451	20.8743	24.077	12.062	0	0.00890738	0.0126892
方法 5	1.24123e-034	1.39699e-030	6.85751e-030	0.00356	17.544	22.398	16.8992	21.0721	13.8431	0	0.00835723	0.0120048

3）Rastrigrin 函数 从表 2 可以看出使用线性调整的加速因子 PSO 在优化此函数时没有明显的改进，甚至比使用固定加速因子稍差，这与文献中所得的结论是基本一致的。但是用凹函数方法和笔者提出的策略则较固定权值策略有一定的提高。图 5 表明了凹函数策略在解决非线性的、多峰值分布的优化问题中，早期的收敛速度较快，但是在算法后期（1000 代后）由于过于相信“社会”知识导致过早收敛，收敛精度固定权值方法相比没有明显的优势；反之，非对称反余弦策略由于在算法后期保留一定的加速因子权值，局部搜索能力较强，故总体性能较优。

4）Griewank 函数 从表 2 和图 6 可看出，笔者提出的加速因子设置策略有较好的寻优能力，实验过程中多次达到全局最优值 0，反观凹函数方法，虽然初期搜索效率很高，但是在算法后期由于粒子缺乏多样性很容易过早收敛，这也表明了非对称反余弦方法在该问题上有较快的收敛速度和最好的收敛精度。

（下转第 16 页）

$$\boldsymbol{D}^{(1)}=\begin{bmatrix} 0 & 5 & 16 & 19_2 & 12 \\ 20 & 0 & 36_1 & 14 & 32_1 \\ \infty & 30_5 & 0 & 20 & 18 \\ \infty & \infty & \infty & 0 & \infty \\ 32_2 & 12 & 48_{21} & 9 & 0 \end{bmatrix} \quad \boldsymbol{D}^{(2)}=\begin{bmatrix} 0 & 5 & 16 & 19_2 & 12 \\ 20 & 0 & 36_1 & 14 & 32_1 \\ 50_{52} & 30_5 & 0 & 20 & 18 \\ \infty & \infty & \infty & 0 & \infty \\ 32_2 & 12 & 48_{21} & 9 & 0 \end{bmatrix}$$

接着根据 $\boldsymbol{D}^{(2)}$ 计算 $\boldsymbol{D}^{(3)}$，由 $\boldsymbol{D}^{(2)}$ 知 $\boldsymbol{D}^{(3)}$ 中需要计算的元素为 d_{14}^3、d_{24}^3、d_{34}^3、d_{54}^3。经计算可知 $\boldsymbol{D}^{(3)}=\boldsymbol{D}^{(2)}$，算法终止。所以 $\boldsymbol{D}^{(2)}$ 中对应的元素 d_{ij}^2 的值即为节点 v_i 和节点 v_j 之间的最短路长值，对应元素下标为经过的节点。由此得所有两节点间的最短路径与最短路长值如表 1。

采用优化的矩阵算法计算例 1，最多需要计算 3 个矩阵，最短路计算的同时记下使路径变短的节点，并在矩阵中对应的元素用下标标注更简单直接。在计算的过程中，不和顶点对直接相连的插入点不能有效的使路长变短，因此在对距离矩阵的计算过程中没有必要对每个元素进行计算，大大减少了计算量。因此优化的矩阵算法计算量小，计算效率高，切实有效。

表 1 所有两节点间的最短路径与最短路长值

节点	最短路径	路长
1，2	①－②	5
1，3	①－③	16
1，4	①－②－④	19
1，5	①－⑤	12
2，1	②－①	20
2，3	②－①－③	36
2，4	②－④	14
2，5	②－①－⑤	32
3，1	③－⑤－②－①	50
3，2	③－⑤－②	30
3，4	③－④	20
3，5	③－⑤	18
5，1	⑤－②－①	32
5，2	⑤－②	12
5，3	⑤－②－①－③	48
5，4	⑤－④	9

[参考文献]

[1] 翁龙年，亢耀先．运筹学［M］．北京：人民邮电出版社，1988.
[2] 张蕾．矩阵方法求赋权图中最短路的算法［J］．西北大学学报，2004，34（5）：527～530.
[3] 任平安．无向网络中最短路径的标记与减少计算量的方法［J］．纺织高校基础科学学报，2001，14（1）：48～50.
[4] 胡运权．运筹学教程［M］．北京：清华大学出版社，2003.
[5] 钱颂迪．运筹学［M］．北京：清华大学出版社，1990.
[6] 谢政，李建平．网络算法与复杂性理论［M］．长沙：国防科技大学出版社，1995.

[编辑] 洪云飞

（上接第 4 页）

4 结 语

PSO 算法的早期收敛非常快，在算法初期应在保证粒子多样性的基础上，尽快让算法进入局部搜索，才能获得更好的求解效率；而在算法后期则粒子应保留一定的搜索能力，避免过早收敛。笔者介绍的动态非线性调整加速因子的算法，通过在搜索初期加强 PSO 模型中的“认知部分”以增加粒子群内的多样性；在搜索后期加强“社会部分”以加快粒子向全局最优点的收敛速度，从而改善算法的收敛精度。试验结果也表明了新的加速因子调整设置 PSO 算法对单峰及部分多峰函数的收敛性具有明显的改进，并且较原先的设置具有更强的全局搜索能力，说明了笔者提出的策略拥有更强的追踪粒子群中全局最优粒子的能力。

[参考文献]

[1] Ratnaweera A, Halgamuge S. Self-organizing hierarchical particle swarm optimizer with time-varying accceleration coefficients［J］. Evolutionary Computation, 2004, 8 (3): 240～255.
[2] 陈贵敏，贾建援，韩琪．粒子群优化算法的惯性权值递减策略研究［J］．西安交通大学学报，2006，40（1）：53～61.
[3] 冯翔，陈国龙，郭文忠．粒子群优化算法中加速因子的设置与试验分析［J］．集美大学学报，2006，11（2）：146～151.

[编辑] 洪云飞